Subject: Re: [PATCH 0/16] Pid namespaces
Posted by Sukadev Bhattiprolu on Thu, 12 Jul 2007 03:19:37 GMT
View Forum Message <> Reply to Message

Pavel Emelianov [xemul@openvz.org] wrote:
| sukadev@us.ibm.com wrote:
| > Pavel Emelianov [xemul@openvz.org] wrote:
| > | This is "submition for inclusion" of hierarchical, not kconfig
| > | configurable, zero overheaded ;) pid namespaces.
| > |
| > | The overall idea is the following:
| > |
| > | The namespace are organized as a tree - once a task is cloned
| > | with CLONE_NEWPIDS (yes, I've also switched to it :) the new
| >
| > Can you really clone() a pid namespace all by itself ?
| > copy_namespaces() has the following:
| >
| >
| > if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
CLONE_NEWUSER)))
| > return 0;
| >
| > doesn't it mean you cannot create a pid namespace using clone() unless
| > one of the above flags are also specified ?
| >
| > unshare_nsproxy_namespaces() has the following correct check:
| >
| > if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
| > CLONE_NEWUSER | CLONE_NEWPIDS)))
| > return 0;
|
| I have lost a couple of hunks when I splitted the patch :(
| That's the correct version, cap_set fix and the renamed CLONE_ flag.
|
| --- ./include/linux/sched.h.fix 2007-07-06 11:09:33.000000000 +0400
| +++ ./include/linux/sched.h 2007-07-10 13:48:19.000000000 +0400
| @@ -26,7 +26,7 @@
| #define CLONE_NEWUTS 0x04000000 /* New utsname group? */
| #define CLONE_NEWIPC 0x08000000 /* New ipcs */
| #define CLONE_NEWUSER 0x10000000 /* New user namespace */
| -#define CLONE_NEWPIDS 0x20000000 /* New pids */
| +#define CLONE_NEWPID 0x20000000 /* New pids */
|
| /*
| * Scheduling policies
| --- ./kernel/capability.c.fix 2007-07-06 11:09:33.000000000 +0400
| +++ ./kernel/capability.c 2007-07-10 13:50:16.000000000 +0400

```
| @@ -103,7 +103,7 @@ static inline int cap_set_pg(int pgrp_nr
|   int found = 0;
|   struct pid *pgrp;
|
| - pgrp = find_pid(pgrp_nr);
| + pgrp = find_pid_ns(pgrp_nr, current->nsproxy->pid_ns);
|   do_each_pid_task(pgrp, PIDTYPE_PGID, g) {
|    target = g;
|    while_each_thread(g, target) {
| --- ./kernel/fork.c.fix 2007-07-06 11:09:33.000000000 +0400
| +++ ./kernel/fork.c 2007-07-10 13:48:13.000000000 +0400
| @@ -1267,7 +1267,7 @@ static struct task_struct *copy_process(
|     __ptrace_link(p, current->parent);
|
|    if (thread_group_leader(p)) {
| -   if (clone_flags & CLONE_NEWPIDS) {
| +   if (clone_flags & CLONE_NEWPID) {
|      p->nsproxy->pid_ns->child_reaper = p;
|      p->signal->tty = NULL;
|      p->signal->pgrp = p->pid;
| @@ -1434,7 +1434,7 @@ long do_fork(unsigned long clone_flags,
|    else
|     p->state = TASK_STOPPED;
|
| -  nr = (clone_flags & CLONE_NEWPIDS) ?
| +  nr = (clone_flags & CLONE_NEWPID) ?
|     pid_nr_ns(task_pid(p), current->nsproxy->pid_ns) :
|      pid_vnr(task_pid(p));
|
| --- ./kernel/nsproxy.c.fix 2007-07-06 11:09:33.000000000 +0400
| +++ ./kernel/nsproxy.c 2007-07-10 13:48:13.000000000 +0400
| @@ -132,7 +132,8 @@ int copy_namespaces(unsigned long flags,
|
|   get_nsproxy(old_ns);
|
| - if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWUSER)))
| + if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
| +     CLONE_NEWUSER | CLONE_NEWPID)))
|    return 0;
|
|   if (!capable(CAP_SYS_ADMIN)) {
| @@ -184,7 +185,7 @@ int unshare_nsproxy_namespaces(unsigned
|   int err = 0;
|
|   if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
| -          CLONE_NEWUSER | CLONE_NEWPIDS)))
| +          CLONE_NEWUSER)))
|    return 0;
```

```
|
|   if (!capable(CAP_SYS_ADMIN))
| --- ./kernel/pid.c.fix 2007-07-06 11:09:33.000000000 +0400
| +++ ./kernel/pid.c 2007-07-10 13:48:19.000000000 +0400
| @@ -523,7 +523,7 @@ struct pid_namespace *copy_pid_ns(unsign
|   BUG_ON(!old_ns);
|   get_pid_ns(old_ns);
|   new_ns = old_ns;
| - if (!(flags & CLONE_NEWPIDS))
| + if (!(flags & CLONE_NEWPID))
|    goto out;
|
|   new_ns = ERR_PTR(-EINVAL);
```

I applied the above patch and the following patch to your
16 patches you first sent.

```
--- ./fs/proc/root.c.procfix    2007-07-10 13:52:08.000000000 +0400
+++ ./fs/proc/root.c    2007-07-10 15:23:20.000000000 +0400
@@ -111,7 +111,7 @@ void __init proc_root_init(void)
        err = register_filesystem(&proc_fs_type);
        if (err)
                return;
-       proc_mnt = kern_mount(&proc_fs_type);
+       proc_mnt = kern_mount_data(&proc_fs_type, &init_pid_ns);
```

My x86_64 system boots fine but crashes as below, when I run my
'pidns_exec' test with a simple program that prints getpid(), getppid()
etc of the process in the child pid ns.

Pls see

http://www.geocities.com/sukadevb/Pidspace/2.6.22-rc6-mm1-pavel-1.tgz

for the patches I currently have applied and let me know if I need more
on top.

And see

http://www.geocities.com/sukadevb/Pidspace/Test1/

for the test programs. You may need to run the 'mypid-loop.x' script
to repro the crash. The pidns_exec.c program calls clone() with CLONE_NEWPID
and execs the given program (it was included in Patch 0 of the patchset I
posted to Containers).

Suka

login: Unable to handle kernel NULL pointer dereference at 00000000000002fc RIP:
 [<ffffffff802b9e5e>] proc_get_sb+0xfb/0x138
PGD 104d53067 PUD 104d4d067 PMD 0
Oops: 0002 [1] SMP
CPU 2
Modules linked in:
Pid: 3279, comm: pidns_exec Not tainted 2.6.22-rc6-mm1-ovz1 #10
RIP: 0010:[<ffffffff802b9e5e>] [<ffffffff802b9e5e>] proc_get_sb+0xfb/0x138
RSP: 0018:ffff8101029d7d28 EFLAGS: 00010202
RAX: ffff810100651840 RBX: ffff810104461400 RCX: ffff810100651878
RDX: 0000000000000000 RSI: ffffffff806e5460 RDI: 0000000000000238
RBP: ffff810102d886f8 R08: ffff810104461400 R09: ffff810100026000
R10: 0000000000000000 R11: 0000000000000002 R12: ffff8101029c6000
R13: 0000000002000000 R14: ffffffff806ee920 R15: ffff810102088cc0
FS: 00002b0b499ec6f0(0000) GS:ffff81010069c3c0(0000) knlGS:0000000000000000
CS: 0010 DS: 0000 ES: 0000 CR0: 000000008005003b
CR2: 00000000000002fc CR3: 000000010381b000 CR4: 00000000000006e0
DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
DR3: 0000000000000000 DR6: 00000000ffff0ff0 DR7: 0000000000000400
Process pidns_exec (pid: 3279, threadinfo ffff8101029d6000, task ffff81010269e7f0)
Stack: ffff810102088cc0 ffff810102088cc0 00000000ffffffff4 ffffffff806ee920
 ffffffff8065f9d9 ffff8101029c6000 0000000002000000 ffffffff80287164
 00000000000000d0 ffff8101029c6000 ffffffff806e5460 ffff8101029c6000
Call Trace:
 [<ffffffff80287164>] vfs_kern_mount+0x4f/0x8b
 [<ffffffff802b9cf4>] pid_ns_prepare_proc+0x13/0x2e
 [<ffffffff80245be3>] copy_pid_ns+0xd7/0x164
 [<ffffffff8024af34>] create_new_namespaces+0xde/0x192
 [<ffffffff8024b0aa>] copy_namespaces+0x4b/0x85
 [<ffffffff802347e2>] copy_process+0xcb4/0x1439
 [<ffffffff8020bbee>] system_call+0x7e/0x83
 [<ffffffff8023556a>] do_fork+0x6c/0x1e7
 [<ffffffff8020bf07>] ptregscall_common+0x67/0xb0

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers